Fig. 1

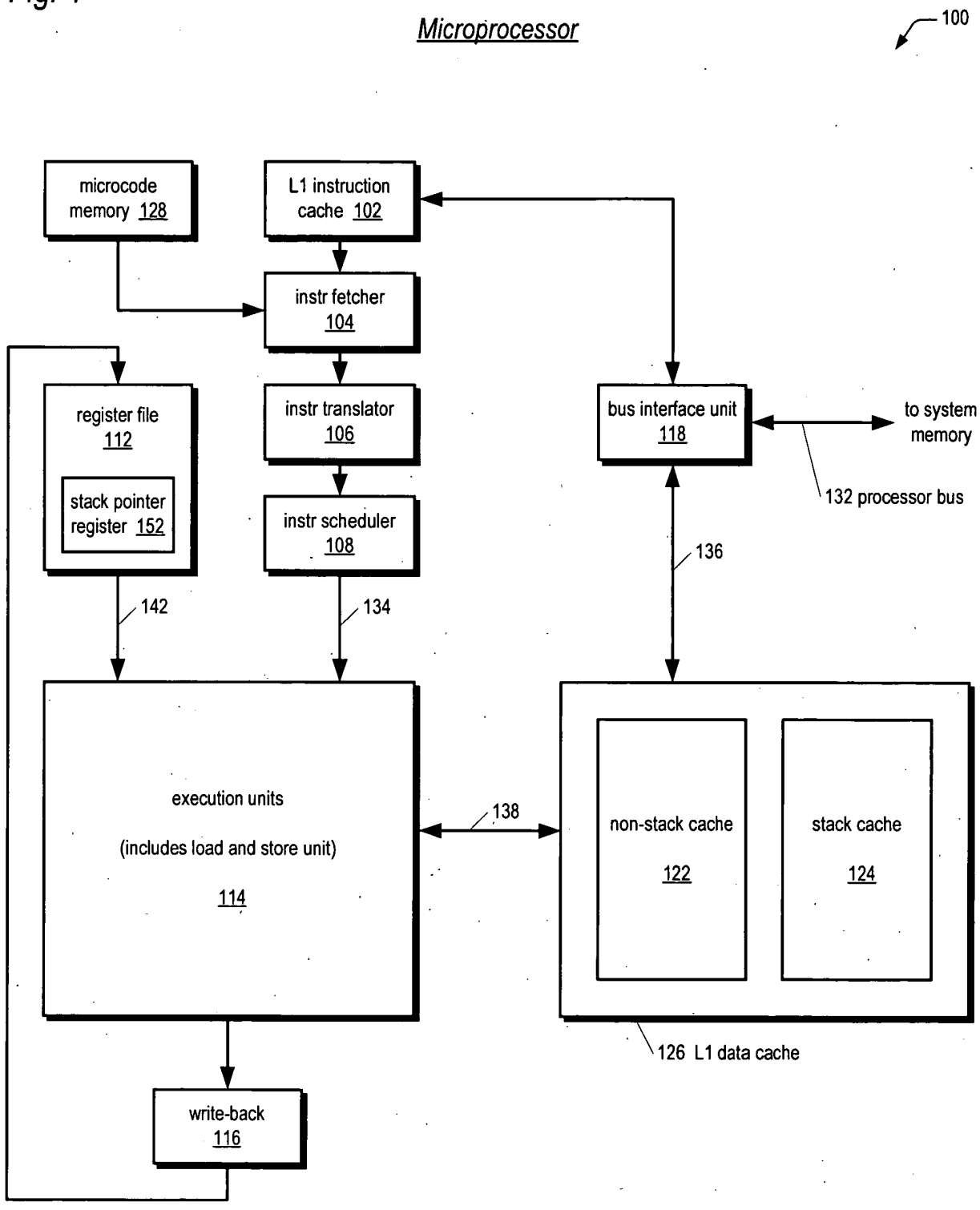

Microprocessor
100
microcode memory 128
L1 instruction cache 102
instr fetcher 104
register file 112
stack pointer register 152
instr translator 106
instr scheduler 108
bus interface unit 118
to system memory
132 processor bus
136
142
134
execution units
(includes load and store unit)
114
138
non-stack cache
122
stack cache
124
126 L1 data cache
write-back 116

Fig. 2

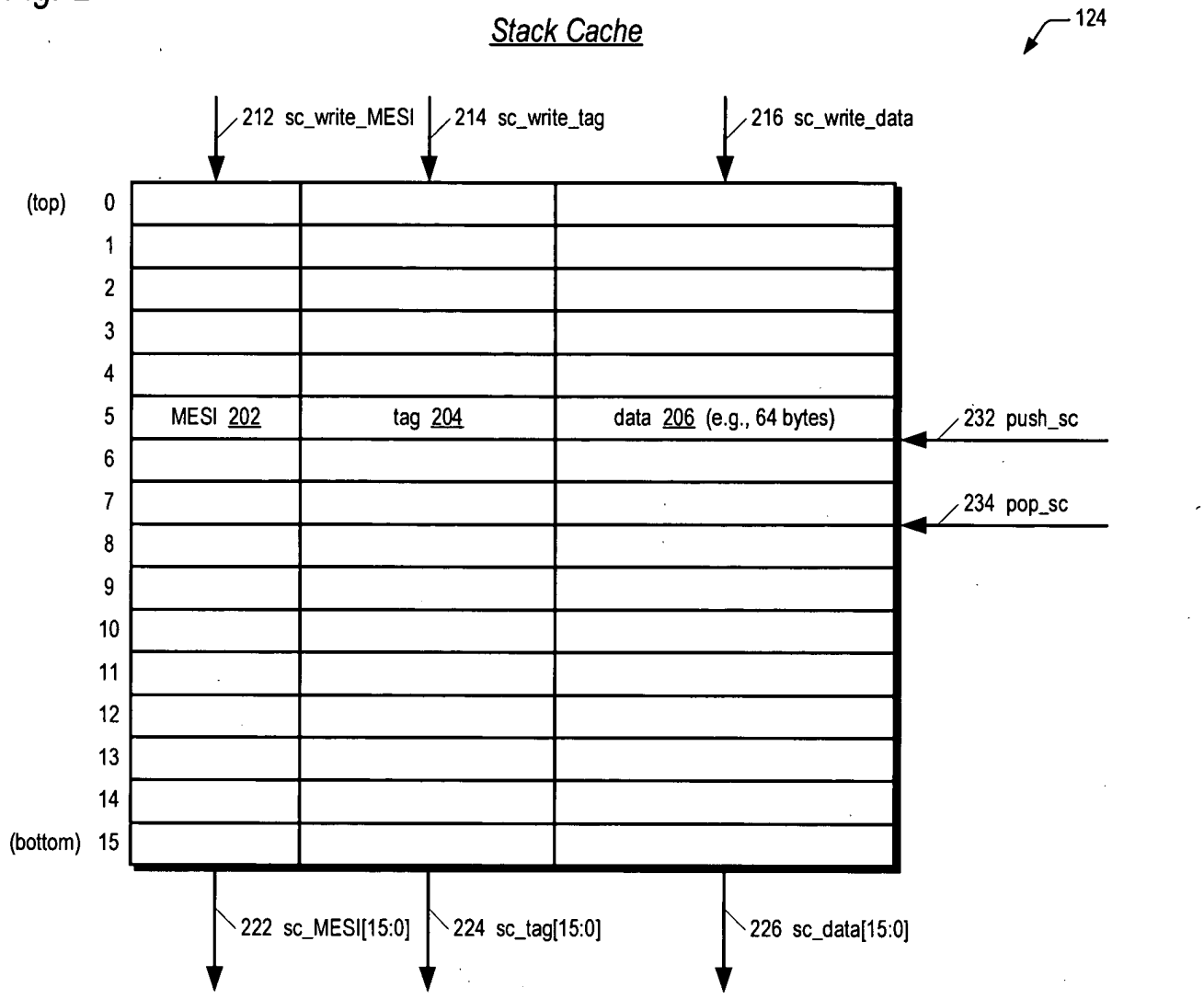

Stack Cache
124
212 sc_write_MESI
214 sc_write_tag
216 sc_write_data
(top)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
(bottom)
15
MESI 202
tag 204
data 206 (e.g., 64 bytes)
232 push_sc
234 pop_sc
222 sc_MESI[15:0]
224 sc_tag[15:0]
226 sc_data[15:0]

Fig. 3

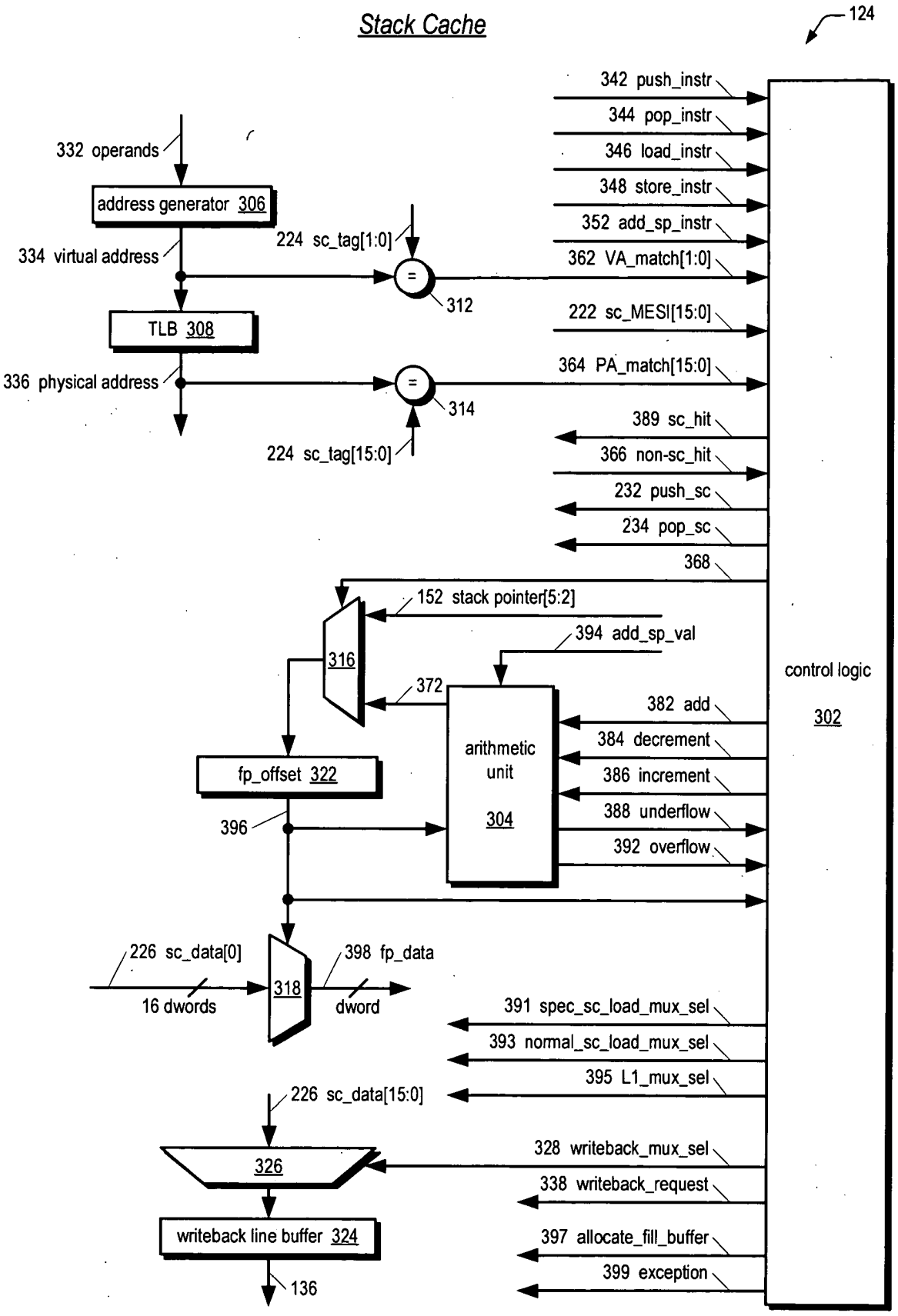

Stack Cache
124
342 push_instr
344 pop_instr
346 load_instr
348 store_instr
352 add_sp_instr
362 VA_match[1:0]
222 sc_MESI[15:0]
364 PA_match[15:0]
389 sc_hit
366 non-sc_hit
232 push_sc
234 pop_sc
368
332 operands
address generator 306
334 virtual address
224 sc_tag[1:0]
312
TLB 308
336 physical address
314
224 sc_tag[15:0]
152 stack pointer[5:2]
394 add_sp_val
316
372
382 add
384 decrement
386 increment
388 underflow
392 overflow
arithmetic unit 304
fp_offset 322
396
control logic 302
226 sc_data[0]
16 dwords
318
398 fp_data
dword
391 spec_sc_load_mux_sel
393 normal_sc_load_mux_sel
395 L1_mux_sel
226 sc_data[15:0]
326
328 writeback_mux_sel
338 writeback_request
writeback line buffer 324
397 allocate_fill_buffer
399 exception
136

Fig. 4

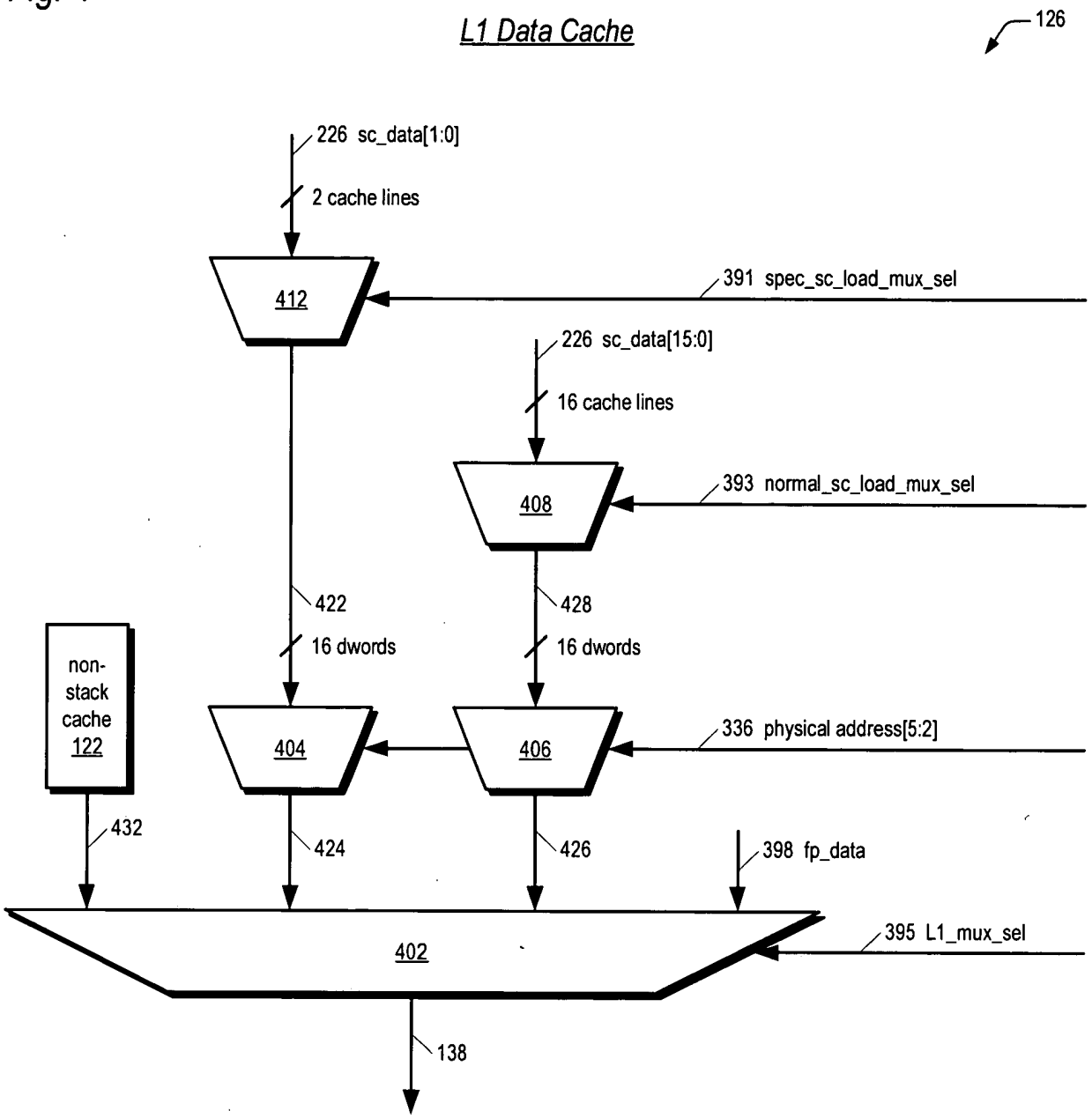

L1 Data Cache
126
226 sc_data[1:0]
2 cache lines
412
391 spec_sc_load_mux_sel
226 sc_data[15:0]
16 cache lines
408
393 normal_sc_load_mux_sel
422
16 dwords
428
16 dwords
non-stack cache 122
404
406
336 physical address[5:2]
432
424
426
398 fp_data
402
395 L1_mux_sel
138

Fig. 5

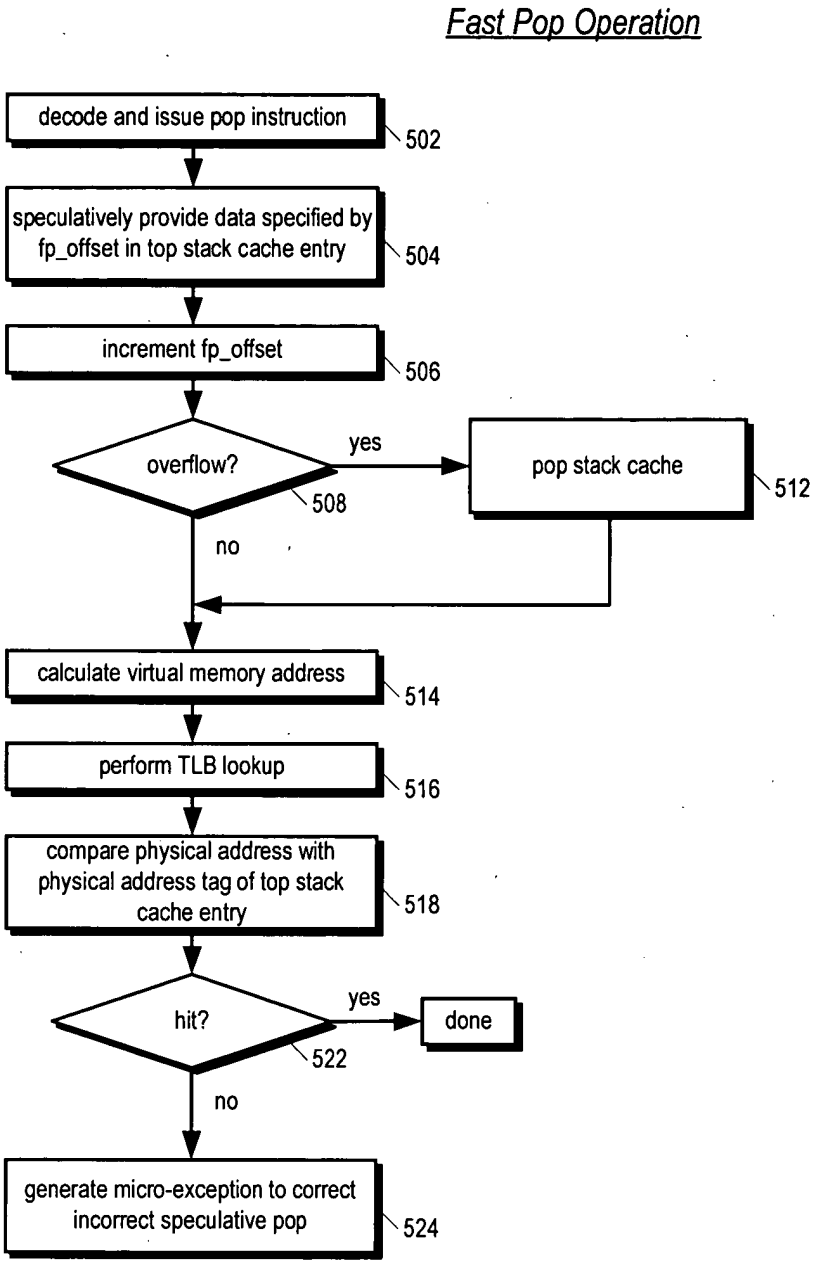

Fast Pop Operation
decode and issue pop instruction
502
speculatively provide data specified by fp_offset in top stack cache entry
504
increment fp_offset
506
overflow?
508
yes
pop stack cache
512
no
calculate virtual memory address
514
perform TLB lookup
516
compare physical address with physical address tag of top stack cache entry
518
hit?
522
yes
done
no
generate micro-exception to correct incorrect speculative pop
524

Fig. 6

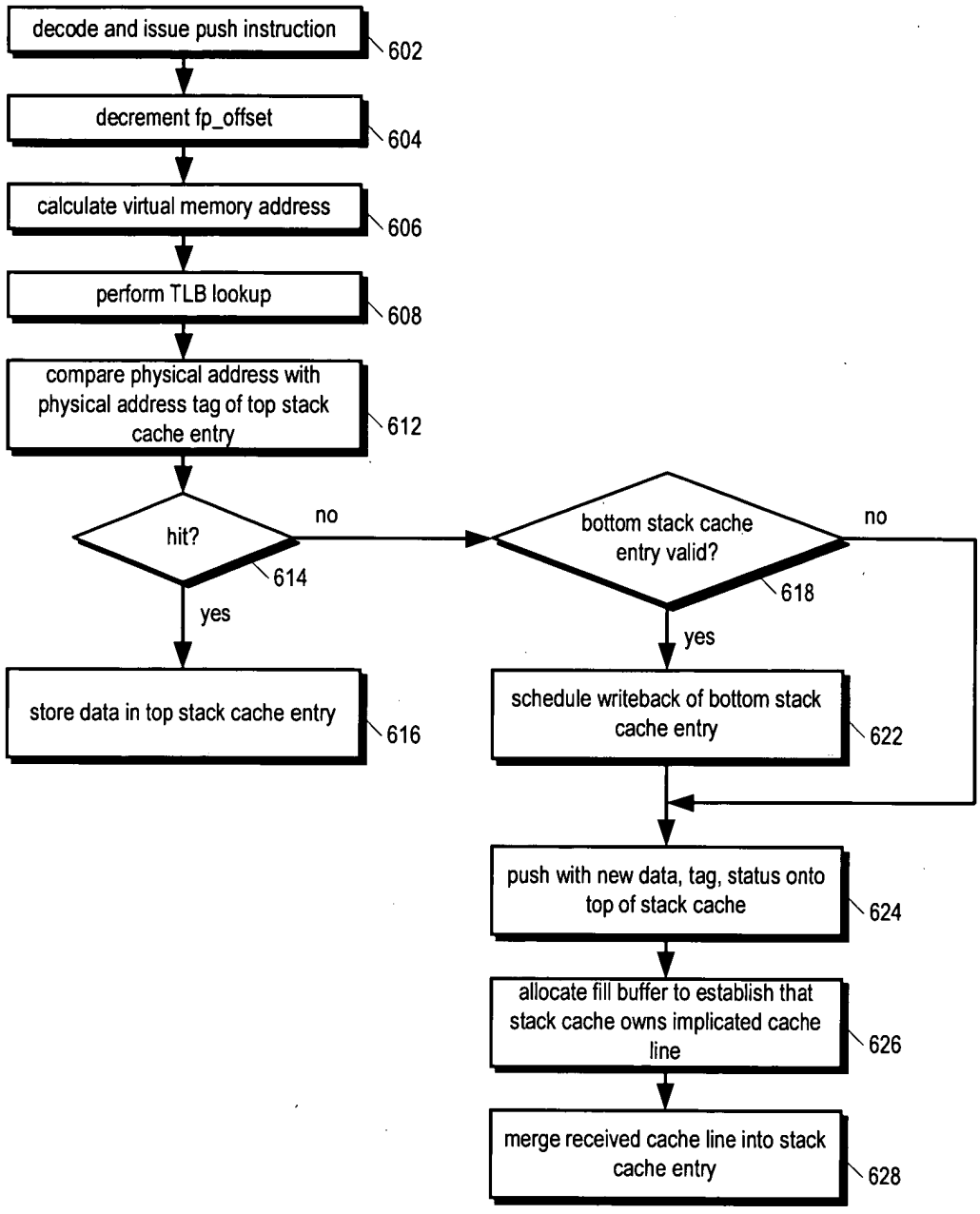

Push Operation
decode and issue push instruction
602
decrement fp_offset
604
calculate virtual memory address
606
perform TLB lookup
608
compare physical address with physical address tag of top stack cache entry
612
hit?
614
no
yes
bottom stack cache entry valid?
618
no
yes
store data in top stack cache entry
616
schedule writeback of bottom stack cache entry
622
push with new data, tag, status onto top of stack cache
624
allocate fill buffer to establish that stack cache owns implicated cache line
626
merge received cache line into stack cache entry
628

Fig. 7

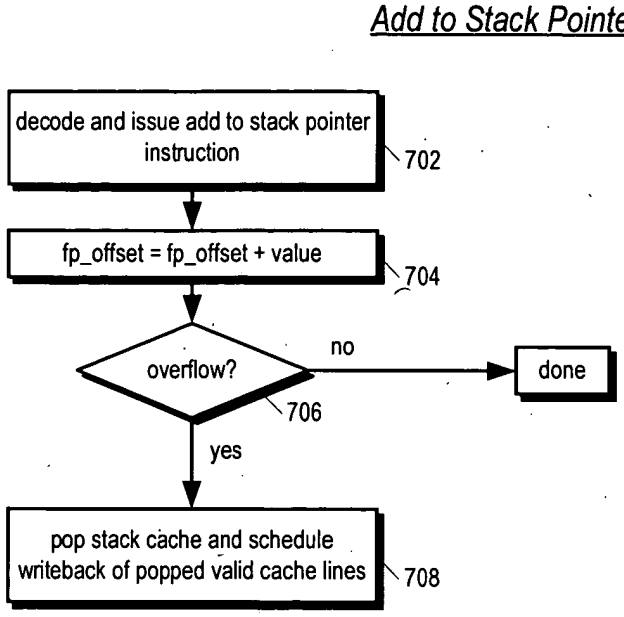

Add to Stack Pointer Operation
decode and issue add to stack pointer instruction
702
fp_offset = fp_offset + value
704
overflow?
706
no
done
yes
pop stack cache and schedule writeback of popped valid cache lines
708

Fig. 8A

Speculative Load from Stack Cache Operation

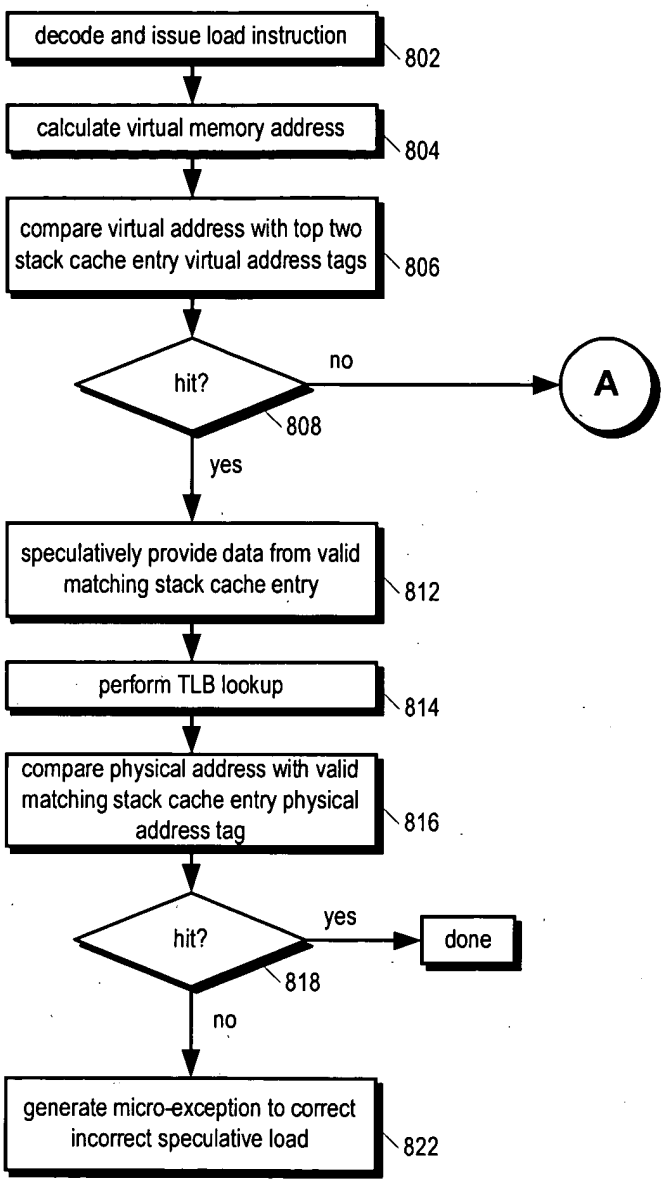


Fig. 8B

Normal Load from Stack Cache Operation

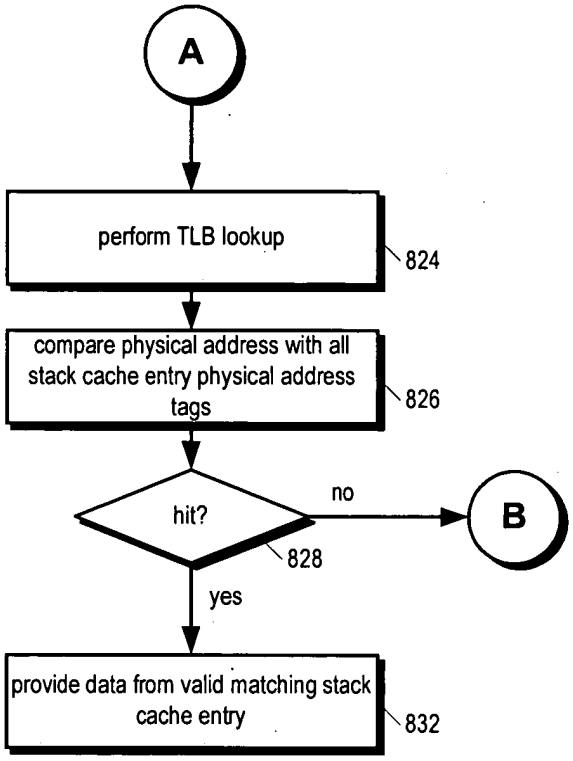


Fig. 8C

Load from Non-Stack Cache Operation

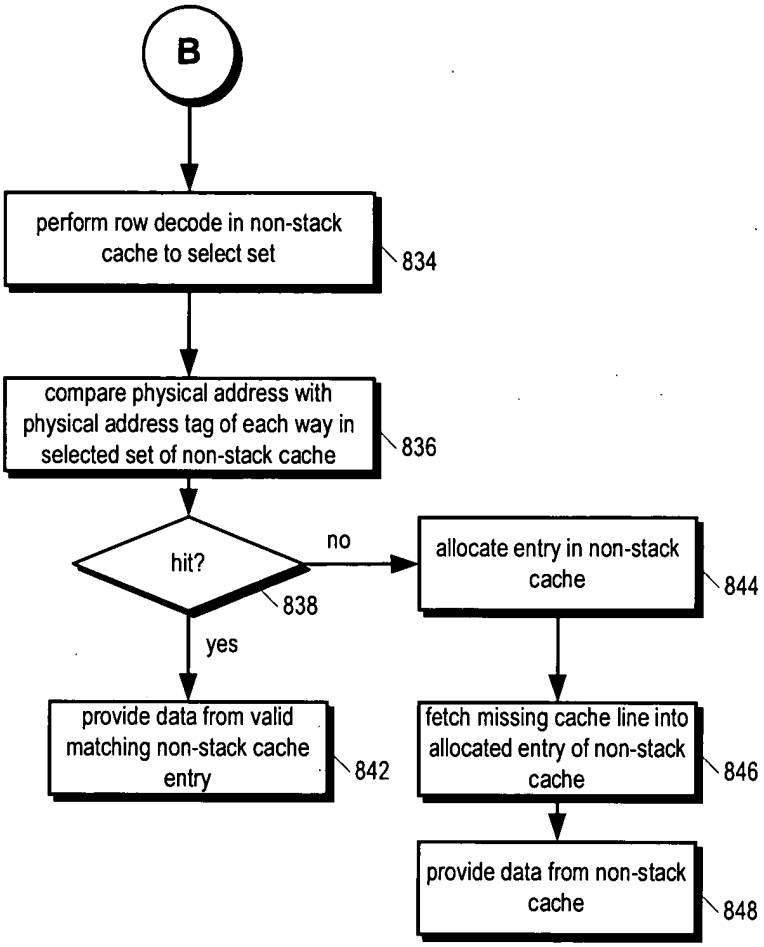


Fig. 9

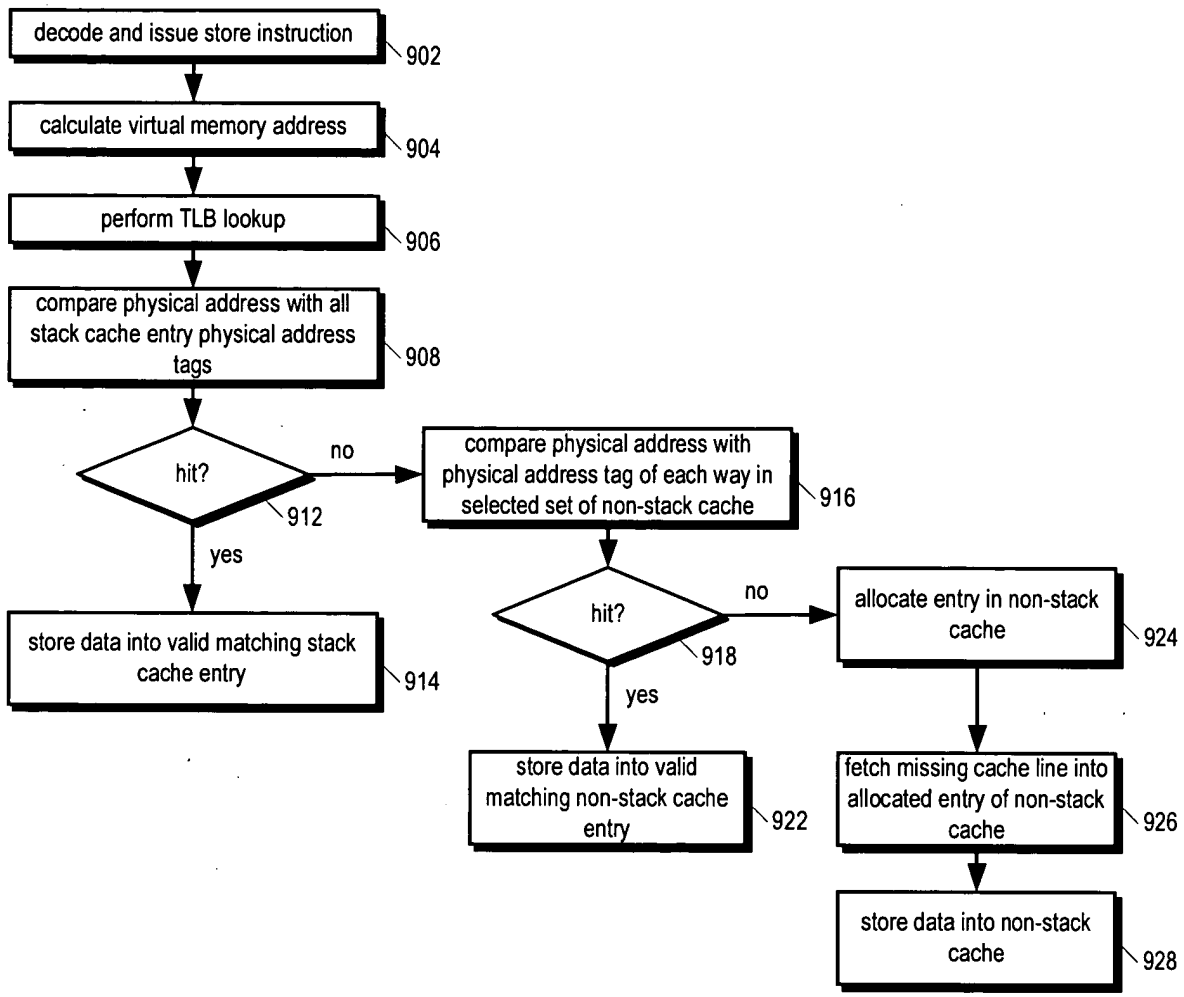

Store Operation
decode and issue store instruction
902
calculate virtual memory address
904
perform TLB lookup
906
compare physical address with all stack cache entry physical address tags
908
hit?
912
no
yes
compare physical address with physical address tag of each way in selected set of non-stack cache
916
store data into valid matching stack cache entry
914
hit?
918
no
yes
allocate entry in non-stack cache
924
store data into valid matching non-stack cache entry
922
fetch missing cache line into allocated entry of non-stack cache
926
store data into non-stack cache
928

Fig. 10 *Fast Pop from Stack Cache Timing*

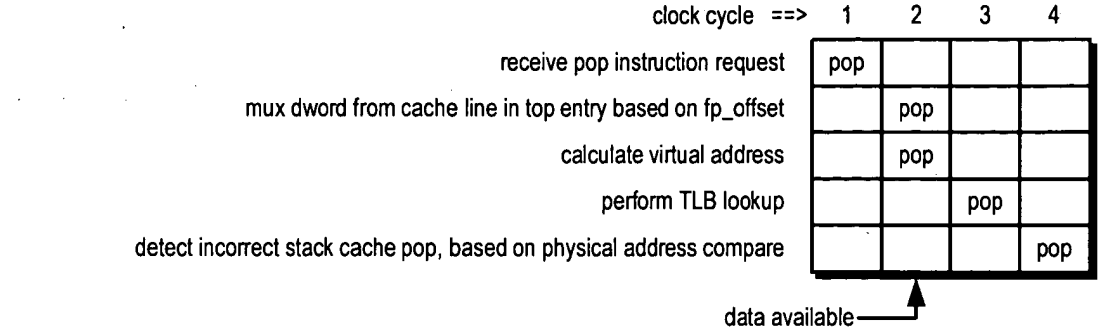


Fig. 11 *Speculative Load from Stack Cache Timing*

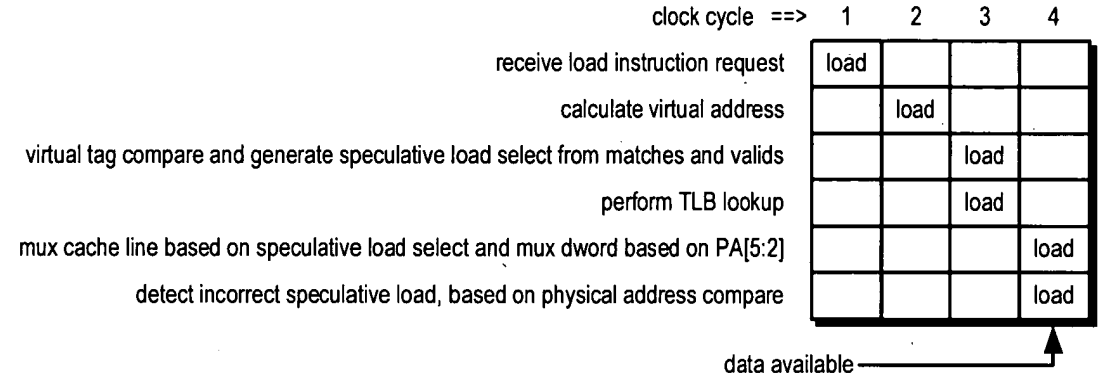


Fig. 12 *Normal Load from Stack Cache Timing*

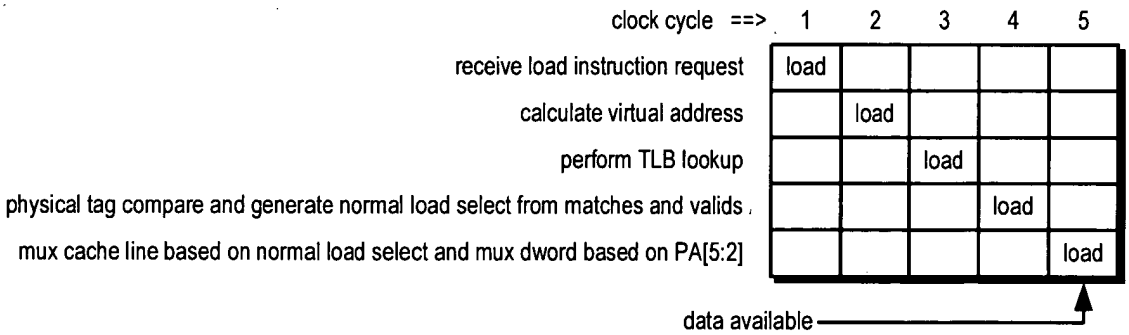


Fig. 13 *Load from Non-Stack Cache Timing*

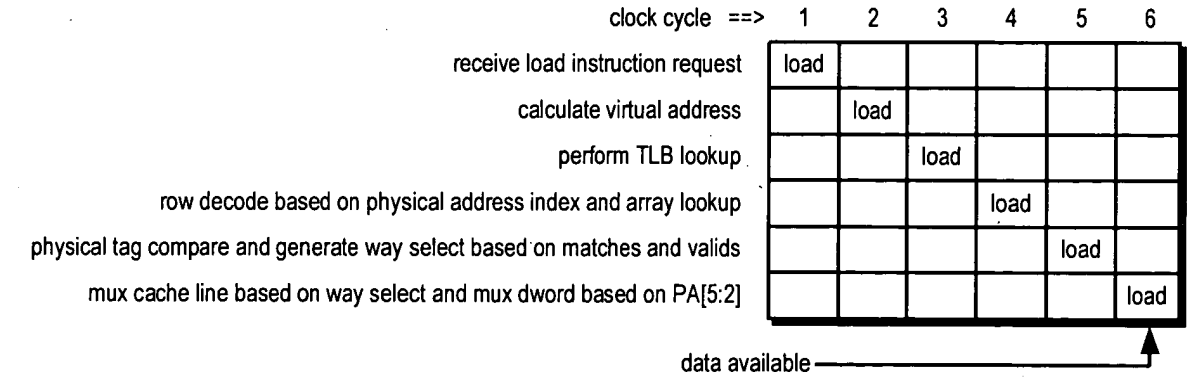


Fig. 14

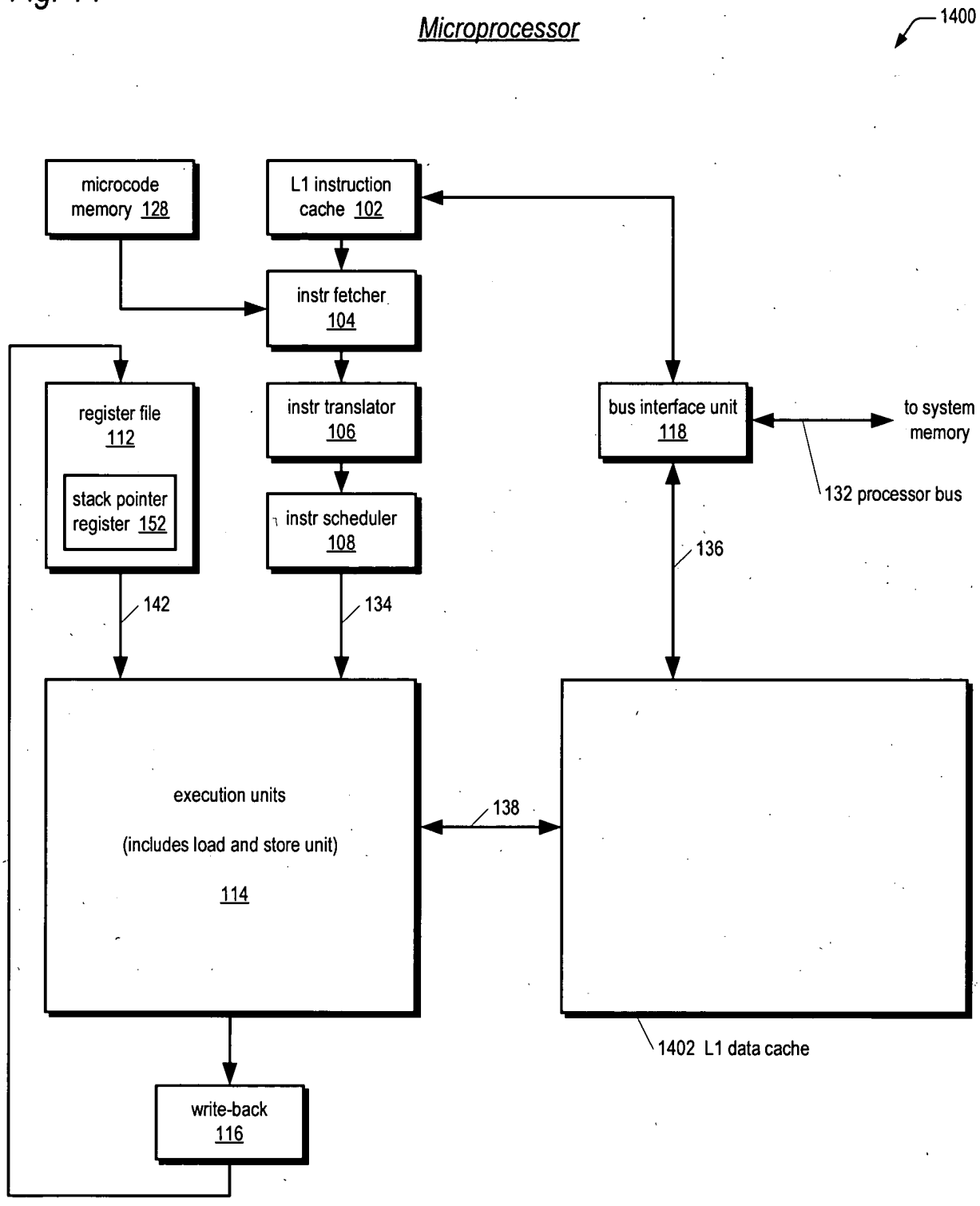

Microprocessor
1400
microcode memory 128
L1 instruction cache 102
instr fetcher 104
register file 112
stack pointer register 152
instr translator 106
instr scheduler 108
bus interface unit 118
to system memory
132 processor bus
136
142
134
execution units
(includes load and store unit)
114
138
1402 L1 data cache
write-back 116

Fig. 15

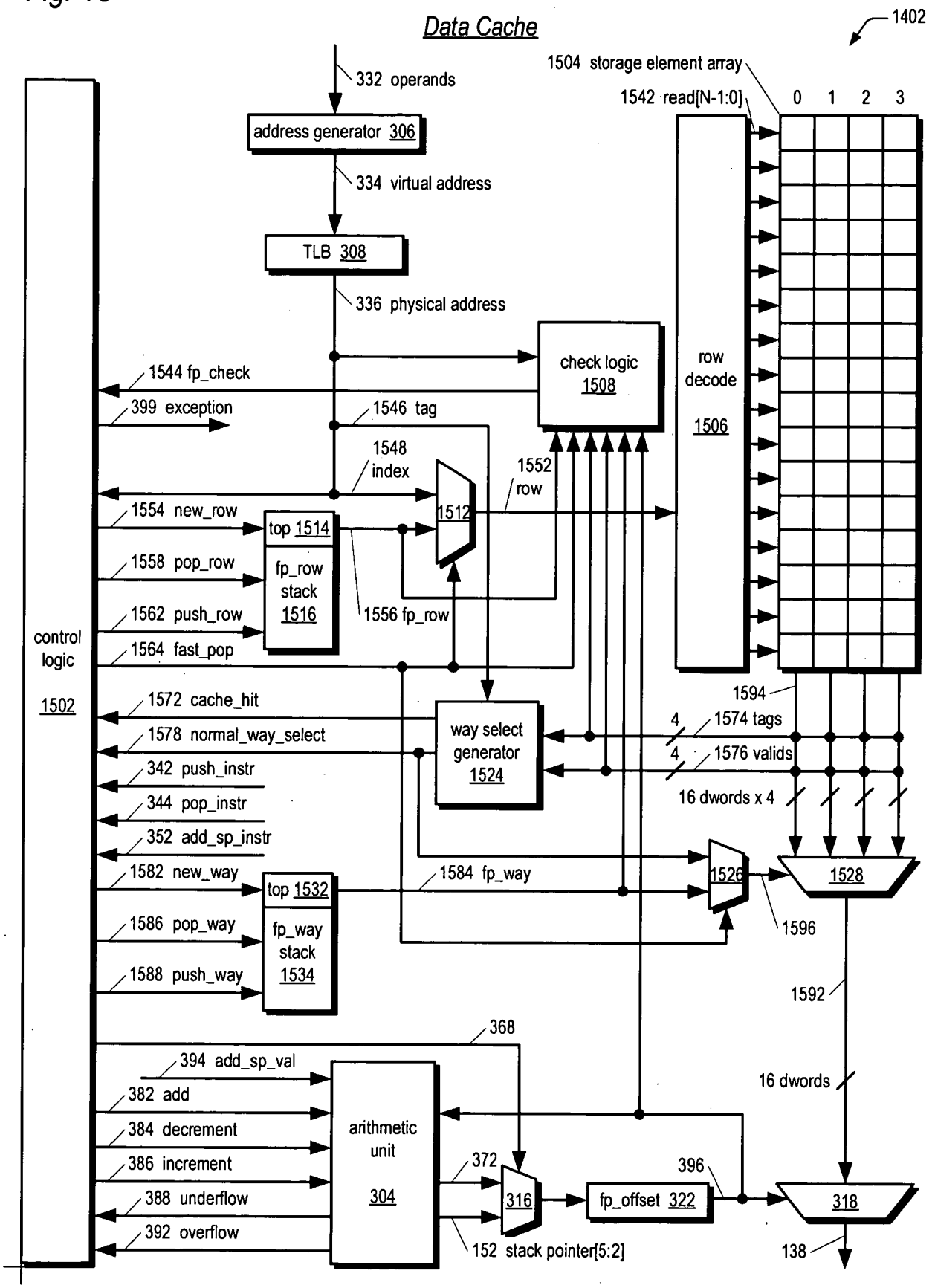

Data Cache
1402
1504 storage element array
1542 read[N-1:0]
0 1 2 3
332 operands
address generator 306
334 virtual address
TLB 308
336 physical address
check logic 1508
row decode 1506
1544 fp_check
399 exception
1546 tag
1548 index
1552 row
1512
1554 new_row
top 1514
fp_row stack 1516
1558 pop_row
1562 push_row
1556 fp_row
control logic 1502
1564 fast_pop
1572 cache_hit
1578 normal_way_select
342 push_instr
344 pop_instr
352 add_sp_instr
way select generator 1524
1594
4
1574 tags
4
1576 valids
16 dwords x 4
1582 new_way
top 1532
fp_way stack 1534
1584 fp_way
1526
1528
1596
1586 pop_way
1588 push_way
1592
368
394 add_sp_val
382 add
384 decrement
386 increment
388 underflow
392 overflow
arithmetic unit 304
16 dwords
372
316
396
fp_offset 322
318
152 stack pointer[5:2]
138

Fig. 16

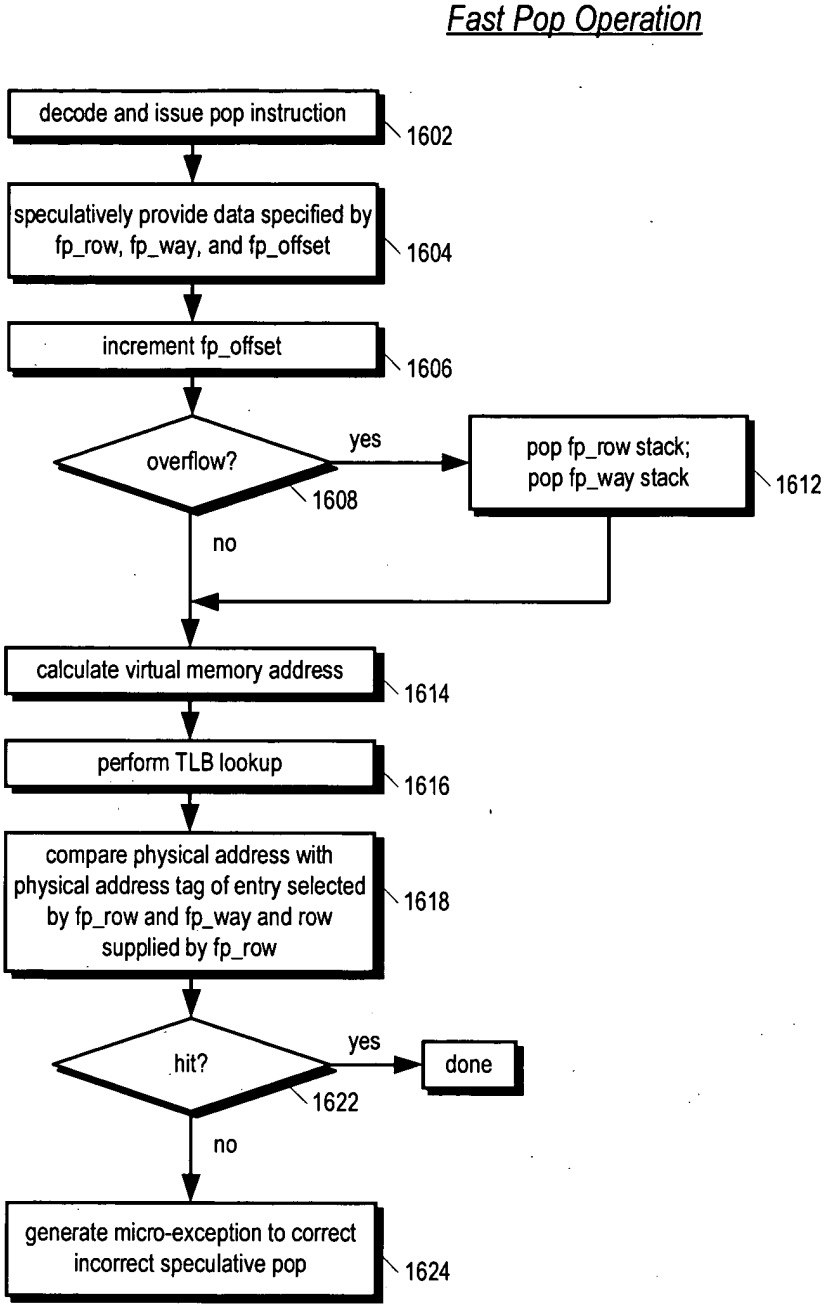

Fast Pop Operation
decode and issue pop instruction
1602
speculatively provide data specified by fp_row, fp_way, and fp_offset
1604
increment fp_offset
1606
overflow?
1608
yes
pop fp_row stack; pop fp_way stack
1612
no
calculate virtual memory address
1614
perform TLB lookup
1616
compare physical address with physical address tag of entry selected by fp_row and fp_way and row supplied by fp_row
1618
hit?
1622
yes
done
no
generate micro-exception to correct incorrect speculative pop
1624

Fig. 17

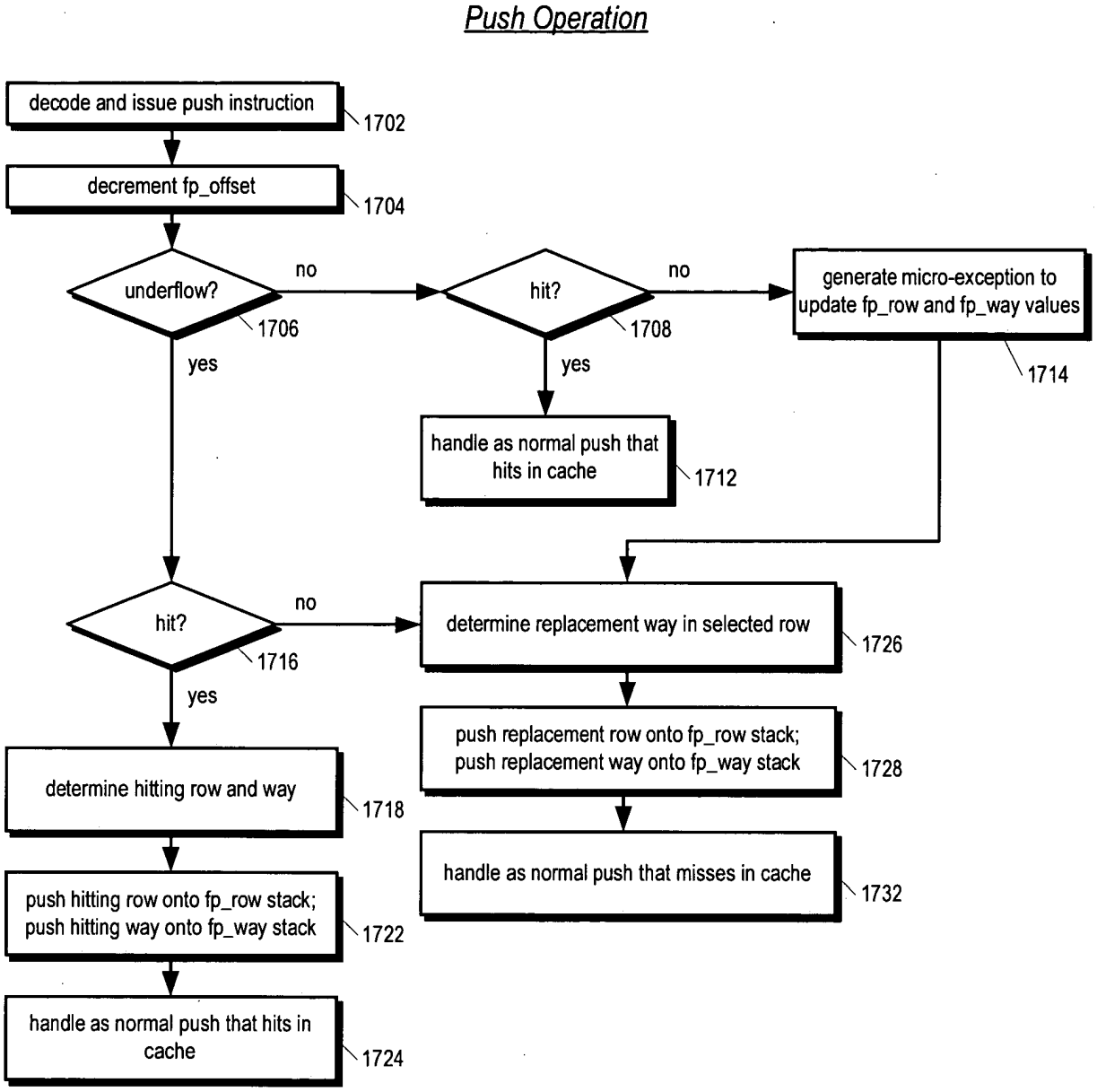

Push Operation
decode and issue push instruction
1702
decrement fp_offset
1704
underflow?
1706
no
yes
hit?
1708
no
yes
generate micro-exception to update fp_row and fp_way values
1714
handle as normal push that hits in cache
1712
hit?
1716
no
yes
determine replacement way in selected row
1726
push replacement row onto fp_row stack; push replacement way onto fp_way stack
1728
handle as normal push that misses in cache
1732
determine hitting row and way
1718
push hitting row onto fp_row stack; push hitting way onto fp_way stack
1722
handle as normal push that hits in cache
1724

Fig. 18

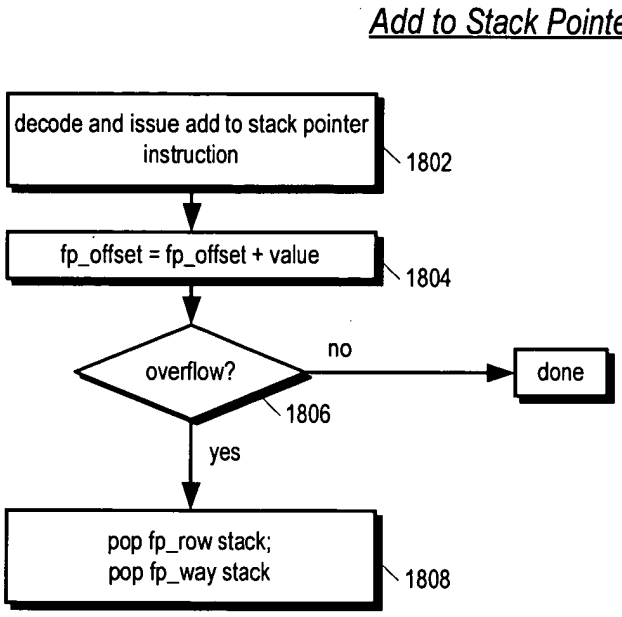
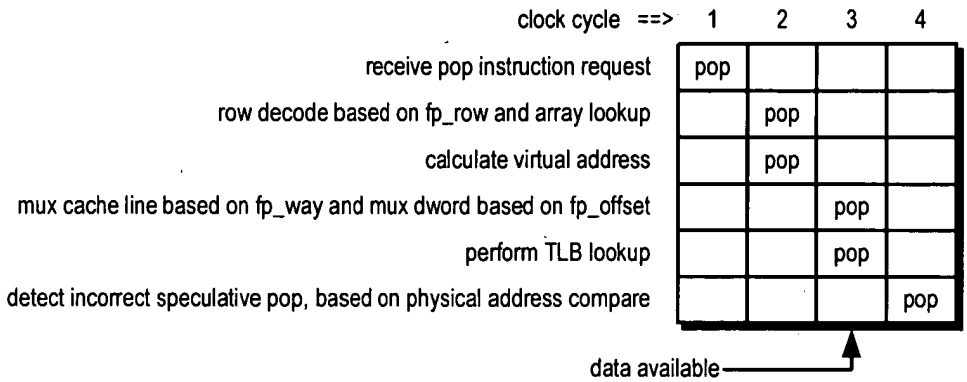

Add to Stack Pointer Operation
decode and issue add to stack pointer instruction
1802
fp_offset = fp_offset + value
1804
overflow?
1806
no
done
yes
pop fp_row stack;
pop fp_way stack
1808

Fig. 19

Fast Pop from Cache Timing